Improving caching efficiency and quality of experience with CF-Dash

Zied Aouini, Tourad Mamadou, Ali Gouta, Anne-Marie Kermarrec, Yannick Lelouedec

Improving caching efficiency and quality of experience with CF-Dash

Zied Aouini
Orange Labs
zied.aouini@orange.com

Mamadou Tourad DIALLO
Orange Labs
mamadoutourad.diallo@orange.com

Ali Gouta
Orange Labs
ali.gouta@orange.com

Anne-Marie Kermarrec
Inria
anne-marie.kermarrec@inria.fr

Yannick Lelouedec
Orange Labs
yannick.lelouedec@orange.com

ABSTRACT

HTTP Adaptive Streaming (HAS) is gradually being adopted by Over The Top (OTT) content providers. In HAS, a wide range of video bitrates of the same video content are made available over the internet so that clients' players pick the video bitrate that best fit their bandwidth. Yet, this affects the performance of some major components of the video delivery chain, namely CDNs or transparent caches since several versions of the same content compete to be cached. In this context we investigate the benefits of a Cache Friendly HAS system (CF-DASH), which aims to improve the caching efficiency in mobile networks and to sustain the quality of experience of mobile clients. Firstly, we motivate our work by presenting a set of observations we made on large number of clients requesting HAS contents. Secondly we introduce the CF-Dash system and our testbed implementation. Finally, we evaluate CF-dash based on trace-driven simulations and testbed experiments. Our validation results are promising. Simulations on real HAS traffic show that we achieve a significant gain in hit-ratio that ranges from 15% up to 50%..

Categories and Subject Descriptors

C.2 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design

General Terms

Design, Performance

Keywords

HAS, Dash, MOS, GPAC, cache, hit-ratio

The research leading to these results has received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n 318398, project eCOUSIN

NOSSDAV 2014 Singapore, Singapore

http://dx.doi.org/10.1145/2578260.2578268

1. INTRODUCTION AND RELATED WORKS

Mobile video is expected to increase 16-fold between 2012 and 2017, accounting for over two-thirds of the world's mobile data traffic by 2017 [11]. This growth in video is accompanied by a key technology trend: the shift toward adaptive streaming over HTTP (HAS). In HAS, the client player dynamically adjusts the video bitrate as a function of the network condition and CPU usage. While this may reduce the playback interruptions, it still adversely affects the user experience with the rise of the number of switching between qualities during the video session [8]. This has also a straight impact on the performance of transparent caches or CDNs, since many bitrates of the same chunk from the same content compete to be cached [5].

Currently, MPEG is working towards a new edition of DASH. This second edition, introduced as SAND [1](Server and Network assisted DASH), aims at giving the network delivery actors such as operators and CDN providers the ability to assist the client player to select the video quality that both clients' players and content delivery actors find it convenient to serve. MPEG has addressed various issues related to the actual HAS version [2], among them was the impact on the existing internet infrastructure such as servers, proxies, caches and CDNs. In this paper, we investigate this impact and propose a solution to tackle this issue. In [4], authors investigated the case of multiple bitrate-adaptive client players sharing a bottleneck link and competing for bandwidth. They focused on scenarios involving a limited number of clients on small scale network. This is because the bandwidth allocation becomes unfair when considering a relative small number of players sharing a LAN like link. Our work is complementary to these works. First, in case of mobile networks the bottleneck is likely to be shifted toward the nodeB or SGSN or GGSN gateways which are shared by tens of thousands of mobile clients. Second, our approach aims at optimizing the network delivery side which in return increases the user experience. In this context we investigate the benefits of a Cache Friendly HAS system (CF-DASH), which aims to improve the caching efficiency in mobile networks by softly limiting access to the highest encoding bitrates in the HAS client adaptation logic without compromising the user engagement and the ability of the HAS client to react adequately in case of adverse network conditions. The main contributions are the following:

- We carry out observations on clients requesting HAS contents in an operational network. Then we identify the profiles that are commonly requested by clients. Then we give insights regarding the user quality perception as a function of the encoding profile. This study aims to identify which profile should we privilege within the delivery infrastructure, we call this profile: *profile-limit*.

- We promote the *profile-limit* within the cache: By default clients experiencing a high bandwidth do not systematically move beyond the profile limit since we assume that the user-engagement is guaranteed at that profile. This makes the large majority of clients experiencing good network conditions to request the same profile (i.e. *profile limit*), hence this raises the probability to this profile to persist longer in the cache.

- We carry out testbed experiments and trace-driven simulations to evaluate CF-Dash. Simulations show that we may achieve a gain in hit-ratio that ranges from 15% up to 50% if properly selecting the profile limit and the cache size.

- Clients still have the option to move beyond the profile-limit. They should manually select profiles higher than the profile limit.

The remainder of this paper is organized as follows. In Section 2 we present the observations made on a large scale mobile network that motivate our work. In section 3, we introduce the CF-Dash system and we describe our testbed implementation. In section 4, we evaluate CF-Dash based on trace-driven simulations and testbed experiments and show that it achieves the goals that we have set. Finally, in section 5, we conclude the paper.

2. MOTIVATION

Our motivation stems from the results of an empirical study that we describe below. This section reports on our main findings.

2.1 Empirical study

In [5] we analyzed a large dataset of HAS sessions collected over a period of 3 months over the French network of a major European mobile operator. The dataset includes 92.595.115 HTTP requests for HAS chunks from 246.913 unique active clients. Each request was logged along with the request timestamp, the size of the chunk, the length of the embedded video segment and the encoding bitrate. Our motivation for the present study is based on the following three observations made in [5].

First, we showed that clients' players often switch between the encoding bitrates. On average the number of transitions during a HAS session falls within the interval [1/6, 1/2] of the total requested chunks per session. In case where a caching system (e.g. CDN or transparent cache) is deployed on the Gi interface of the mobile network, our simulations show that this high switching behavior and video bitrate selection heterogeneity adversely affects the cache hit ratio by 15%.

Profile i	Encoding bitrate (kbps)
Profile 0 (P_0)	$\leqslant 50$
Profile 1 (P_1)	[50-150)
Profile 2 (P_2)	[150-280)
Profile 3 (P_3)	[280-420)
Profile 4 (P_4)	[420-600)
Profile 5 (P_5)	[600-1000)
Profile 6 (P_6)	[1000-2000)
Profile 7 (P_7)	≥ 2000

Table 1: Profiles

Second, the analysis of the encoding bitrates in the collected data led us to classify them into a set of representative ranges, called Profiles (See Table 1): the vast majority of the videos had each of their encoding bitrates belonging to one of these profiles and they did not have two encoding bitrates in the same Profile. Profile 5 was clearly the most frequently requested one over all HAS sessions, while the next most frequently ones were lower profiles; mostly requested at the beginning of the sessions (See Figure 1).

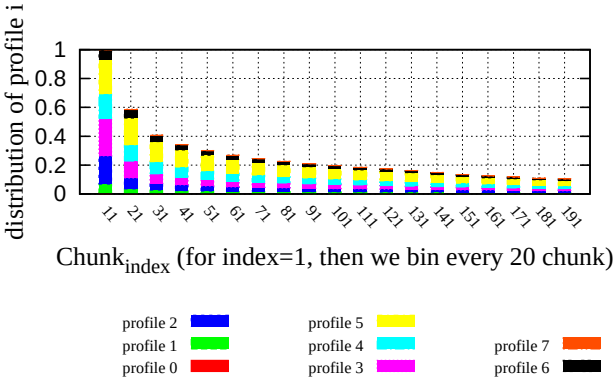


Figure 1: Distribution of requested profiles with respect to the $chunk_{index}$ for catch-up (VoD) contents

Third, we used markov chains to characterize the switching between profiles. We estimated the probability to move from $P_{i\in[0..7]}$ to profile $P_{j\in[0..7];j\neq i}$ (c.f. transition matrix P). We observed that the probability to move from $P_{i=5}$ to $P_{j>5}$ is 22%, then when clients are in $P_{i=6}$, they are more likely to switch to $P_{j=5}$ with a probability equal to 60%. When being in P_7, clients move to P_6 and P_5 with a probability of 92%. All this suggests that when being in P_5, we observe that a significant proportion of transitions are limited within the highest profiles. When being in P_4, around 45% of transitions are made to the upper profiles, and that 48% are made from $P_{i=5}$ to $P_{j=4}$. This intermittently switching between the highest profiles does not necessarily yield improvement of quality of experience. But instead, it adversely affects some network components such as caches.

$$P = \begin{pmatrix} 0 & 0.0997 & 0.2650 & 0.2685 & 0.1784 & 0.1358 & 0.0460 & 0.0066 \\ 0.0053 & 0 & 0.1533 & 0.3482 & 0.2683 & 0.1753 & 0.0464 & 0.0029 \\ 0.0036 & 0.0405 & 0 & 0.4237 & 0.28 & 0.2034 & 0.0456 & 0.0028 \\ 0.0021 & 0.0410 & 0.2496 & 0 & 0.4204 & 0.2486 & 0.0339 & 0.0041 \\ 0.0007 & 0.0192 & 0.1271 & 0.3867 & 0 & 0.4080 & 0.0500 & 0.0079 \\ 0.0009 & 0.0099 & 0.0817 & 0.2005 & 0.4790 & 0 & 0.2016 & 0.0260 \\ 0.001 & 0.0055 & 0.0303 & 0.0739 & 0.1617 & 0.6060 & 0 & 0.1213 \\ 0.0004 & 0.0018 & 0.0078 & 0.0160 & 0.0441 & 0.2266 & 0.7031 & 0 \end{pmatrix}$$

These three observations show that there is a potential opportunity to improve the caching efficiency in the mobile network by favoring one specific profile in the HAS adaptation logic. without compromising the user engagement and the ability of HAS clients to react adequately in case of network congestion and adverse conditions on the server and client devices.

2.2 QoE evaluation

In this section we show results of a subjective quality perception tests made by the operator who provided the dataset we described in the previous section. Tests were conducted with five ranges of quality from bad 1 to excellent [4,5]. The purpose of the experiment was to correlate the encoding profiles with the perceived quality. Type of videos used for the tests fits well the type of VoD contents analyzed in [5]: News, Animation and Film. The tests were conducted with 27 subjects. Subjects were invited to watch video contents twice: on smart phone, then on Tablet. The resolution description and results of the average Mean Opinion Score (MOS) of each profile are reported in Table 2.

Profile	Video resolution	User perception (MOS)
Profile 1 (P_1)	176 x 144	bad (1)
Profile 2 (P_2)	280 x 160	bad (1.2)
Profile 3 (P_3)	320 x 180	medium (2.2)
Profile 4 (P_4)	400 x 224	good (3.3)
Profile 5 (P_5)	480 x 270	good (3.8)
Profile 6 (P_6)	640 x 360	Excellent (4)
Profile 7 (P_7)	1024 x 576	Excellent (4.5)

Table 2: MoS

In [10], authors suggests that a minimum guaranteed MOS equal to 3 is required to ensure an acceptable service quality for any connected user. This requirement is guaranteed with P_4 and P_5.

Based on these considerations, we observe that defining a wide range of encoding profiles (such the content providers do actually) would let clients' players switch often between the highest profiles. Whereas we observe that the user engagement is guaranteed at a specific profile. Next, we present CF-Dash that leverages these observations to improve caching efficiency.

3. CACHE FRIENDLY-DASH

In this part, we introduce CF-Dash adaptation logic and details our implementation and testbed setup.

3.1 Cache Friendly-Dash in a nutshell

In CF-Dash, we intentionally promote one specific profile within the cache. This is achieved by pushing clients' players to request one specific and commonly requested profile, we call it *profile-limit*. By doing so, the *profile-limit* will be populated within the cache and clients will be more likely to be served from the cache. From the network standpoint (i.e. Core network), our approach reduces the aggregated number of switching between qualities experienced by all clients and hence ensures stability of clients' players. In CF-Dash, even though clients experience a high bandwidth, by default clients' players never ask for profiles above the *profile-limit*. The *profile-limit* is chosen in such a way we guarantee a good user-engagement. Hence switching to the highest profiles would be unnecessary and would not affect the user-engagement. Now, for clients who want to move beyond the *profile-limit*; they have to select manually profiles higher than the *profile-limit*. The rationale behind this idea is to prevent clients to turn systematically to the highest profiles when they experience a high bandwidth.

3.2 PoC implementation

In our PoC, we chose GPAC [6] as an Open Source multimedia framework installed on end-terminals, since it incorporates the major DASH standard components. We chose Squid as a proxy cache and Apache as HTTP server. We integrated our proposed changes within the core of GPAC, and carried out testbed experiments to evaluate our approach and validate our implementation.

In CF-Dash, clients' players and cache-servers share information so that the clients' player learns about the *profile-limit*. In MPEG, contributors consider three options to make clients' players and content delivery servers communicate: either by modifying the MPD at some network level (*i.e.* CDN), or by adding new fields in the HTTP header, or by adding a new interface through which clients' players and network components exchange messages. In our PoC, when a client starts a new video session. First, her HTTP-GET request gets forwarded to the proxy-cache server. Then, the proxy-cache is configured so that it does not cache the initialization segment [2] that points to the *profile-limit* but caches all the rest. Squid implements the *X-CACHE* header. Therefore, if the init segment is not cached, the X-CACHE header will contain a miss. The HTTP-Response being sent to the client, the latter parses the header fields and maps the miss into the *profile-limit* candidate. Herein we consider 2 cases:

Leader. This is the case when a new video is being added to the catalogue of VoDs, and a first user (we call it leader) requests that video. This first client in time scale would not be able to identify the *profile-limit*, since none of the init segments is being cached. Therefore, the leader will download all init segments from the origin server and admits that the latest downloaded init segment (highest profile) corresponds to the profile limit.

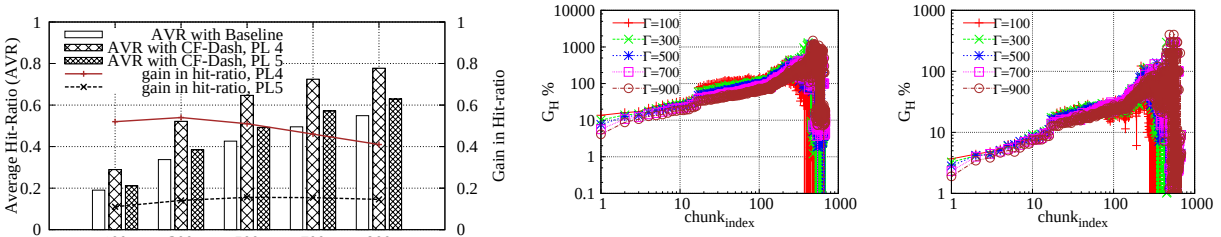
Followers. Followers refer to all users who will successfully identify the *profile-limit* based on the aforementioned mechanism.

3.2.1 Pseudo code

Algorithm 1 depicts the pseudo-code of CF-Dash rate-adaptation logic:

- In Require: *AdaptationSet* and *Representation* are dash standard words naming.*Representation* refers to the encoding profile.

[2]During the connection setup, the clients' player downloads all init segments defined within the Media Presentation Description. Each init segment is associated to one encoding profile. The init segment has the metadata needed by the player to decode the associated profile.



(a) Average and Gain in hit ratio

(b) Average Hit-Ratio per chunk index when setting profile limit to 4

(c) Average Hit-Ratio per chunk index when setting profile limit to 5

Figure 2: *Metrics evaluation: Average hit-ratio, G_H*

- Line 1: *disable switching* becomes true when clients select manually profiles higher than the profile limit.
- Lines [6,8]: DL corresponds to the download speed of the last downloaded chunk. In these lines, we compare the last representation (last requested profile) with the new DL.
- Lines [13,28]: This loop parses all existing profiles and tend to identify the profile that fits the user bandwidth. Conditions 19 and 23, forces the clients' player to not surpass the profile limit whatever the value of DL.
- Line 30: We save the actual representation, so that we can compare it with the next computed DL, when requesting the next chunk.

4. EVALUATION

In this part, we demonstrate through simulations and testbed emulation that CF-Dash adaptation logic improves the cache efficiency on large scale.

4.1 Simulation evaluation

We achieved a trace-driven simulation to evaluate the impact of CF-Dash on caching efficiency. The trace considered in the simulation corresponds to HAS sessions collected over a period of 2 weeks from the dataset introduced in Section 2. Each record in the trace corresponds to one HAS session; it includes the timestamp of the connection setup, the reference of the requested video, the number of requested chunks, and for each video bitrate switching the timestamp and the newly requested Profile. The caching system is deployed at the Gi interface on the mobile network. For the sake of simplicity, we assume that: All chunks are 10 second long. The encoded profiles are selected with respect to the following encoding profiles:
$P_{i\epsilon[0..7]} = [40, 100, 210, 250, 510, 900, 1500, 3500]_{kbps}$. LRU is the content replacement algorithm of the caching system. We evaluate the performance of the cache for different values of the capacity C such as: $C = \Gamma * S$, where S represents a video size of *10 MB*, typically a short video-clip of 5 minute long, and Γ represents the number of videos that the cache sustains. In Figure 2(a), we evaluate CF-Dash with the baseline (i.e. native trace) based on 3 key metrics:

Require: Client, AdaptationSet, Representation, Limit_Profile,
1: **if** (disable_switching = TRUE) **then**
2: return
3: **end if**
4: Go_Up=FALSE
5: DL=compute_download_rate(Client)
6: **if** $(Representation.bandwidth \leqslant DL)$ **then**
7: Go_Up=TRUE
8: **end if**
9: **if** $(DL \leqslant AdaptationSet.Min\_Representation\_Bitrate)$ **then**
10: DL=AdaptationSet.Min_Representation_Bitrate
11: **end if**
12: N=AdaptationSet.total_representations
13: **for** $k = 0 \rightarrow N$ **do**
14: SR=Get_Representation(AdaptationSet,k)
15: **if** $(DL \geq Selected\_Representation.bandwidth)$ **then**
16: **if** (!New_Representation) **then**
17: New_Representation=SR
18: **else if** (Go_Up) **then**
19: **if** $(SR.bandwidth \geqslant New\_Representation.bandwidth) and (k \leq AdaptationSet.Limit\_Profile)$ **then**
20: New_Representation=SR
21: **end if**
22: **else**
23: **if** $(SR.bandwidth \geqslant New\_Representation.bandwidth) and (k \leq AdaptationSet.Limit\_Profile)$ **then**
24: New_Representation=SR
25: **end if**
26: **end if**
27: **end if**
28: **end for**
29: **if** $(disable\_switching = FALSE) and (New\_Representation) and (New\_Representation \neq Representation)$ **then**
30: Representation = New_Representation
31: **end if**

Algorithm 1: CF-Dash: rate adaptation logic

- Average hit-ratio ($H = \frac{requests\ served\ from\ the\ cache}{all\ requests}$): average ratio of requests successfully served from the cache.
- Gain on hit-ratio ($G_H = \frac{H_{CF-Dash} - H_{baseline}}{H_{basline}}$), to assess the gain we may achieve on hit-ratio with CF-Dash adaptation logic with regard to the baseline.
- G_H per $chunk_{index}$.

We observe on Figure 2(a) a significant improvement in the hit-ratio when clients adopt CF-Dash adaptation logic. When the *profile-limit* is set to P_4, we reach more than 40% of G_H. When the *profile-limit* is set to P_5, we still reach a gain of 15%.

In [5], we observed that around 65% of HAS sessions are being aborted from the start when clients experience long delays during the *joining-phase*. Figures 2(b) and 2(c) show that reducing the aggregated number of switching gives more chance to the earlier $chunk_{index}$ to hold in the cache. This is because the earlier chunks compete less with the advanced chunk indexes for whom CF-Dash takes action (i.e. when experiencing a high bandwidth, clients' players converge to request the *profile-limit*). This is important for the *joining-phase* since clients will be served from the cache rather than from the origin server, and since this is the most critical phase that impacts the *user-engagement*.

4.2 Experiments evaluation

4.2.1 Test bed and scenario

The experimental testbed and setup is similar to that described in [7]. We install Gpac on both end-terminals. The configuration of each component of the testbed is as follows:

- The Apache HTTP server hosts a catalogue of 20 videos encoded with respect to profiles used in previous simulations. Each video is hosted jointly with its MPD. Videos are segmented into 12 chunks, 10 second long each one. We skip profile 0, since it is rarely requested (c.f. figure 1).
- The cache proxy is placed between the clients and the origin server. Evaluation tests are carried by varying the cache size capacity C. We keep on the same parameters used in the simulations: $C = \Gamma * S$, where S represents a video size of *12 MB*, which is the average size of one video from our catalogue, and Γ represents the number of videos that the cache sustains. We set the *profile limit* to P_5. We use Dummynet on both clients to emulate the bandwidth variation. We use real world bandwidth variation traces [9] of clients being covered by 3G/HSPA network in Norway. We periodically reproduce network conditions when moving by train from Oslo to Vestby, and the way back (c.f. [3]).

Finally, we simulate 200 clients requesting videos from the catalogue. Content popularity is distributed according to Zipf law with parameter equal to 1.

4.2.2 Results and interpretation

We evaluate the following:

- Profiles distribution: We analyze the proportion of each profile per requested chunk. Our goal is to promote the *profile-limit* within the cache.
- Gain in hit-ratio (G_H)(c.f. Section 4.1).
- Stability of clients' players: We show how CF-Dash reduces the aggregated number of switching during HAS sessions.

Profiles distribution. Most HAS implementations (commercial and open source) agree that the first requested chunk from a video content should be preset in the MPD by the content provider. Usually, the content provider defines the startup profile as the lowest encoding profile, so that to guarantee a smooth playback at the joining-phase. In figure 3, we observe that this holds for Gpac for $chunk_1$. Then, clients decide to move to the profile that fits best their available bandwidth. We observe that the *profile-limit* is largely requested by client-players. Profiles higher than the *profile-limit* (i.e. P_6 and P_7 in our case), are still being requested. This is because leaders do not recognize the *profile-limit* as do the followers, since all init segments are downloaded from the origin server.

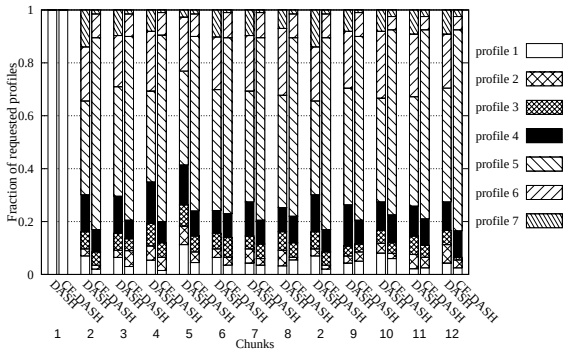
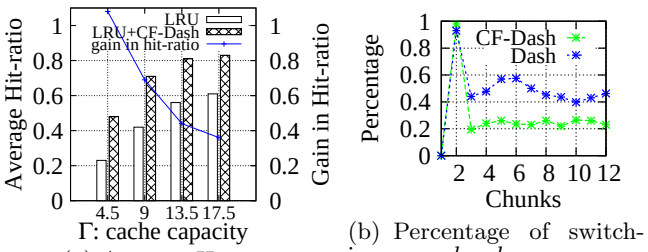


Figure 3: Profiles distribution

Gain in hit-ratio. Figure 4(a) confirms that the G_H significantly increases. When $\Gamma = 4.5$, we observe that CF-Dash adaptation logic allows doubling the performance of the cache with regard to the dash adaptation logic. Then, although the G_H decreases as the cache size increases, we still maintain a promising gain in hit-ratio that reaches around 38% when $\Gamma = 17.5$. Our evaluations demonstrate that CF-Dash significantly improves the hit-ratio.

Stability of client-players. Figure 4(b) shows that most HAS sessions upgrade their video quality when moving from the first to the second chunk of the stream. This is because the emulated bandwidth variation is most of the time higher than the preset profile. With CF-Dash, we decrease the aggregated number of switching over all HAS sessions by



(a) Average Hit ratio

(b) Percentage of switchings per $chunk_{index}$

Figure 4: Evaluation: Hit ratio and stability

20%. Hence, this promotes the *profile-limit* within the cache and increases the opportunity to the followers to be served from the cache.

5. CONCLUSION

In this paper, we investigated the effect of rate-adaptation in HAS on caching-efficiency. Having observed that the QoE is guaranteed from a specific encoding-bitrate, we designed CF-Dash system with the aim to have further control on the rate-adaptation and to promote one specific encoding profile within the cache. In our future work, we will further investigate the ideal profile to be cached and define incentive strategies to encourage clients requesting the same encoding profiles.

6. REFERENCES

[1] `https://datatracker.ietf.org/documents/LIAISON/liaison-2013-11-04-...pdf`.

[2] `http://cmm.khu.ac.kr/korean/download.php?id=808&sid=6a267a6d7b2f771ab06f776dc67a21ba`.

[3] `http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/`.

[4] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 19–24. ACM, 2013.

[5] A. Gouta, D. Hong, A.-M. Kermarrec, Y. Lelouedec, et al. Http adaptive streaming in mobile networks: characteristics and caching opportunities. *MASCOTS2013*, 2013.

[6] J. Le Feuvre, C. Concolato, and J.-C. Moissinac. Gpac: open source multimedia framework. In *Proceedings of the 15th international conference on Multimedia*, pages 1009–1012. ACM, 2007.

[7] C. Mueller, S. Lederer, and C. Timmerer. A proxy effect analyis and fair adatpation algorithm for multiple competing dynamic adaptive streaming over http clients. In *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1–6. IEEE, 2012.

[8] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Spatial flicker effect in video scaling. In *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*, pages 55–60. IEEE, 2011.

[9] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Commute path bandwidth traces from 3g networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 114–118. ACM, 2013.

[10] A. Saul. Wireless resource allocation with perceived quality fairness. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1557–1561. IEEE, 2008.

[11] S. I. ULC. Sandvine global Internet phenomena Report - 1h2012. Technical report, 2012.